DE VINCI
INNOVATION
CENTER

Data Structure and Algorithms

Workshop Sheet

Warm-up **3**
O.O.P 3
Vec2 3
Shapes 3
Event Based 3
Recursion 4
Essential Data Structures **4**
Stack 4
Array 4
Linked List 5
Hash Map 5
Problems 6
Complexity **7**
Graphs **7**
Getting started 7
Applications 8
(Very) Hard Graph Problems 8
Trees **8**
Discovery 8
Manipulating trees 9
Problems 10
Problems **11**

1. Warm-up

a. O.O.P

i. Vec2

1. Implement a Vec2 class, representing a 2 dimensional vector, you have to implement methods to compute the L2 norm of the vector.
2. Add methods to support addition subtraction and multiplying by a scalar. Make these operations in-place, make them return a new Vec2
3. Create a Referential class representing the base in which this vector is. Create a class BasedVec2 which extends from Vec2.
4. What is the problem if you add a Referential to the Vec2 class ?
5. Override the add, sub methods in BasedVec2 to throw an error if operation on 2 vectors in different bases is performed. Make these new methods call the ones from vec2.
6. Make a method in the Referential class to project a vector from another Referential in this Referential, effectively changing the vector's base.

ii. Shapes

1. Using Matplotlib, find a way to draw a line between two points. Then multiple lines on the same graph.
2. Write a Shape class that takes a list of Line (You can also create a Line class), that can accept new lines (e.g with a accept(line)) and can draw them all and display a matplotlib window with a show() method.
3. Make a Square, Rectangle and Line classes that can be constructed with an origin, and the different elements required to build them. (Square has origin, size. Rectangle has origin, width, length)
4. Create a Circle class. Where should you call the plt.plot code ? Modify Shape to accept something else than Lines.
5. Create a Face class with a circle head, eyes and a smile :) Print multiple faces :):):)

b. Event Based

Event based programming is widely used in video games and in the web domain. The goal is to create a Event Based local web server using the http.server built-in Python module.

1. Use the code snippet available here to create a simple python web server. Call this file with python and go to http://127.0.0.1:8000

2. Create a MyHandler class which extends http.server.SimpleHTTPRequestHandler override the do_GET method to print something in the console, then call the super do_GET method
3. Make the server stop on the third page load

c. Recursion

1. Write a program that calculates the sum of a list, using recursion
2. Using recursion, write a program that converts any integer to a string in a base between 2 and 9
3. Write a program, using recursion to sum all the elements of a list even with in-depth lists (Example f([1, [1, 2], [[1], [2,3]]) = 10)
4. Write a recursive to put a number to a certain power

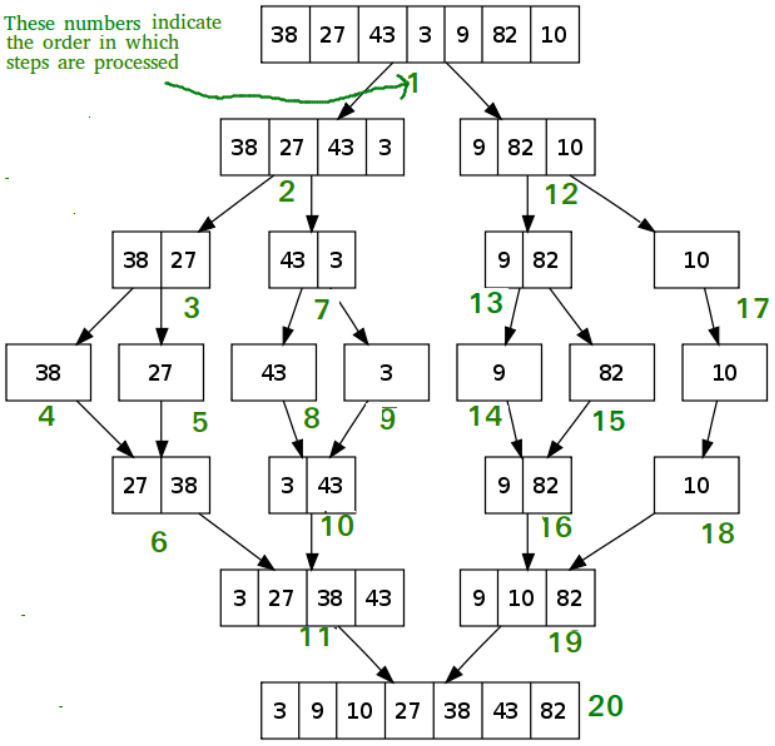
2. Essential Data Structures

a. Stack

1. In Python, write a Stack class that has a add, pop and size method (use __len__ as an alias for size)
2. In C, create a Stack struct, determine what it should contain to be as general as possible, write stack and unstack methods.
3. In any language, write a program to check if a given string has balanced parenthesis. (Eg: “((())()())” is balanced, “)()(“, “(()” are not)
4. Write a calculator that accepts elements in Postfix notation. (E.g: Instead of 2+1, you would input 2 1 +. And instead of (2+1)*4 you would input 2 1 + 4 *)

b. Array

1. Write a function that finds the smallest value in an array
2. Write a function that returns the index of the smallest element in the array
3. Write a function that finds duplicate values in an array. (Assume there is only one duplicate)
4. Write a function that moves all the 0 of an array at the end.
5. Write a recursive function that sorts an array using the Merge Sort method (see below)



6. Write a program to find two elements in an array whose sum is equal to a given number.
7. Given two ordered arrays, write a function that merges the two arrays while maintaining the order.

c. Linked List

1. Implement a Linked List in C (with struct) or Python (with a class). Implement the methods to get the n-th element in the list, insert, remove, add an element at the end of the list and get the size of the list.
2. Write a function to check if a given element is in the list
3. Write a double linked list: each node has access to its next and previous element. Write the insert, delete methods for double linked list
4. Write a program to display a linked list in reverse order. Write a function that reverses the order of the list in place.

d. Hash Map

1. Write a function to sort a Python dictionary
2. Write a function merge two dictionaries
3. Write a function that takes a string and returns a dictionary in which each key is a letter and value the number of times the letter appears in the string
4. Write a program that finds common keys in two dictionaries

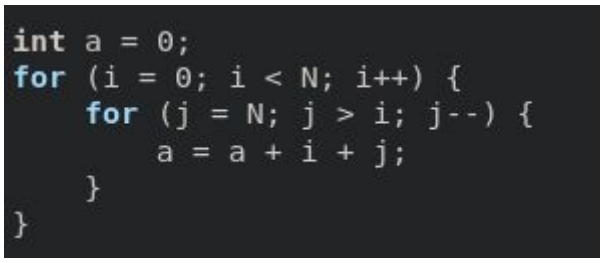
5. Write a program that take two dictionaries and merges them and sum the values of the keys that are in common
6. Write a program to save a dictionary to a file and to load a dictionary from a file

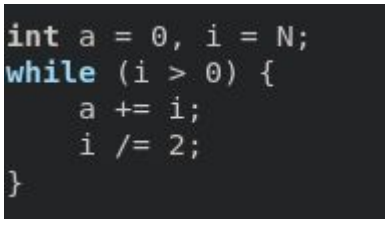
e. Problems

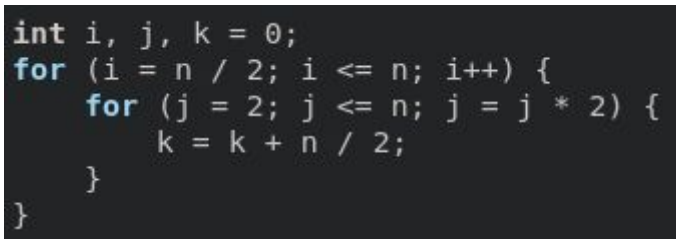
1. Write a function to encode a string in morse code
2. Write a function that take two arrays of integers and returns True if the two arrays combined would form a consecutive sequence (even if unordered)
3. Given an unordered array find the largest possible consecutive sequence in an array (E.g: [100, 2, 1, 50, 3] -> [1, 2, 3])

3. Complexity

Calculate the complexity of these algorithms:

1. 
```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

2. 
```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

3. 
```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

4. What is the complexity of the Merge sort algorithm you wrote earlier ?
5. Write a recursive function to compute the fibonacci sequence. What is the complexity of this function? Write an iterative function that has better complexity.

4. Graphs

a. Getting started

1. Create a graph node class and implement methods to connect/disconnect nodes
2. Make a method to print out the graph layout
3. Make a method to transpose an adjacency matrix to a graph node, make a method that does the opposite
4. Modify the class to make an oriented graph, modify the connect method
5. Modify the class to make a weighted graph, modify the connect method to specify the weight
6. The Dijkstra algorithm is an algorithm to find the shortest path between two nodes in a graph. Implement this algorithm using the structures you just created
7. A* is another form of algorithm to find the shortest path between two nodes in a graph. Implement it in a 2D plane. You first have to find a way to model your graph and represent walls.

b. Applications

Some examples of how graphs are applied to everyday software.

1. State Machine: State machines are oriented graphs, with each edge being an action to perform. State Machines are a subset of Automaton, and are used to model a lot of real world problems, but are also used in video games for animation.
    - Write a state machine that has a default position and accepts actions, which update the state machine. Using actions such as “time passed”, “key pressed”, “key released”, “character touches ground” explain how to make a state machine to render character animation.
    - Optional: You can implement this state machine using PyGame and a simple listener.
2. Friend Graph: How would you model friendship the way it works on facebook ?
3. GPS: How would you model a roadway for a GPS application ? How would you find the shortest path between your location and destination ?

c. (Very) Hard Graph Problems

A set of very hard problems, go as far as possible, don’t get discouraged if you can’t solve them: some of them are current research subjects.

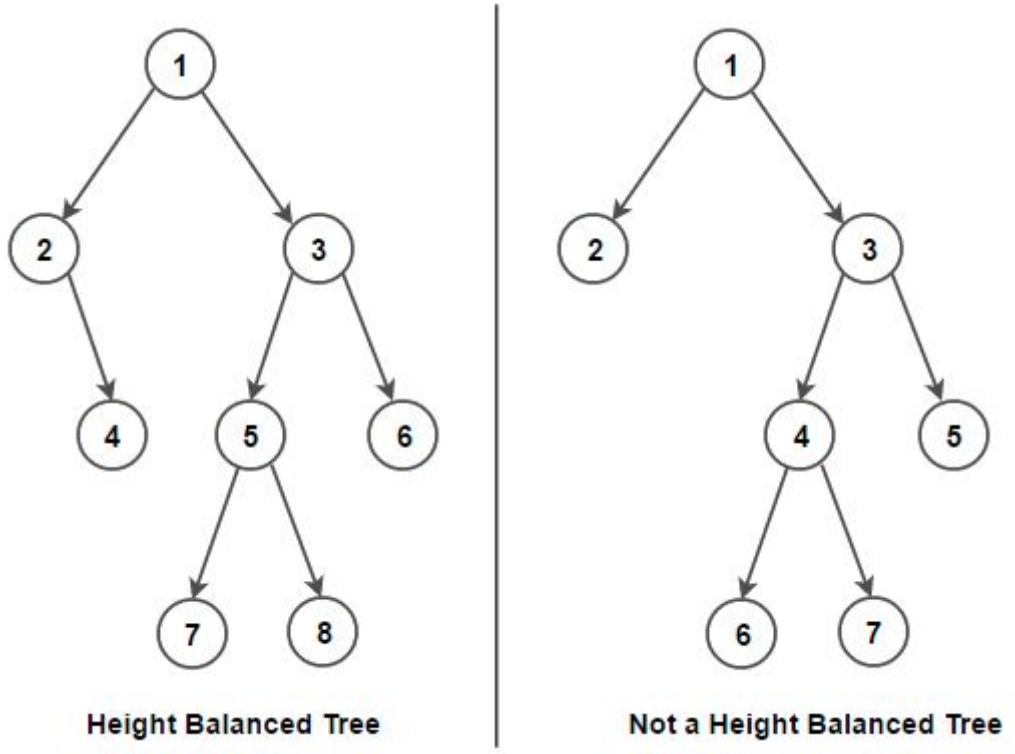
- Traveling salesman: Given a graph, find the cycle with the minimum cost (weight cost)
- Clique: Given a graph, find the largest subset of vertices that are fully connected
- Independent Set: Given a graph, find the largest subset of vertices that are not connected

- Vertex Cover: Given a graph, find the smallest subset of vertices such as any edge in the entire graph contains at least one vexter in that subset

5. Trees

a. Discovery

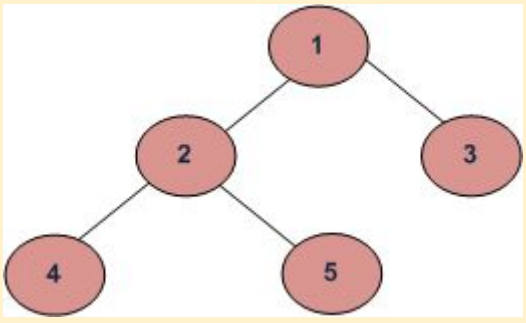
1. Trees and Graph can be implemented in very similar ways, but try to make a Tree and TreeNode classes. Add insert, delete methods in the same saw you did for graphs.
2. Using the Graph tools you made earlier, write a function that determine if a graph is a tree
3. Write a special structure for Binary Trees in which you can access the left and right children easily.
4. Write a function to find the depth of a given tree
5. A **Balanced Binary Tree (BST)** is a tree in which the depth of the two subtrees of every node never differ by more than 1. Eg:



6. Write a function that checks if a binary tree is balanced
7. Write a function to balance an unbalanced binary tree

b. Manipulating trees

- Write a function that print values of a binary tree in a specific order



Eg; here Write 3 functions that write the values in the specified order when supplied this example graph:

infix	4 2 5 1 3
prefix	1 2 4 5 3
postfix	4 5 2 3 1

- **Kd-Trees** is a space segmentation technique very useful to find points in space that are close to each other. Given a set of points (2D), write a function that outputs the kd-tree of this set of points.

c. Problems

- **Minimum Spanning Tree** is a subset of a graph that is acyclic and has the minimum weight sum. Write a function that, given a graph outputs its minimum spanning tree using Prim's algorithm.
- **Huffman** Tree is a structure used for data compression. (For instance, the .zip files use Huffman coding). Given a string of data, output its compressed value using Huffman coding.
    - Step 1: Make the frequency table of your message
    - Step 2: Build the Huffman tree
    - Step 3: Encode the input
- Binary Trees are also used in **database indexing**. Make an example table, with a username, an age and en email.
    - First, represent this table with a linked list. How would you look up a user ?
    - Add a random **but unique** index (integer) to all elements. How can the lookup time on the newly created index be improved ?
    - How can the lookup time be improved on the other elements ? (see the Python hash function)

6. Problems

- Connect 4

Connect 4 (Puissance 4) is a popular game as well as a popular computer science exercise. Write a connect four that can display in the console. The numbers of rows should be configurable.

- Tetris

Tetris is a popular game and its rules are very straightforward. It is however not a suitable game for the console. Use this script to get started with PyGame.
Your tetris game should accept inputs to rotate a piece, make pieces fall down automatically, detect a game over and remove lines.